03 | 问题设计（上）：三大原则理清面试考察方向

2021-04-10 四火

进入课程 >

讲述：四火

时长 19:57　大小 18.27M

你好，我是四火。

相信你在学习了第 02 讲的面试计划之后，心中对于它已经有了更深入的理解。

我们在面试计划中，谈到了规划合适的面试内容，来帮助我们获取面试重点所对应的数据。而在技术面试中，技术问题就是这个面试内容的直接体现。

我们也重点提到了，这个过程中，考察角度、考察重点、考察内容，三者要遵从一致性原则。因此，如果技术问题没有设计正确，对于候选人的考察，就不可能靠谱。

那么接下来的两讲，我就来说说，具体怎么去设计“合适”的技术问题。这一讲我会先从反面介绍几个糟糕的技术问题典型，再从正面讲讲技术问题设计的原则；下一讲我会介绍

一些技术问题的设计技巧，以及一些实践中的注意点。

相信在学习完这两讲之后，你对怎样设计技术问题就能做到心中有谱了。

糟糕的技术问题

首先，让我们来看几个典型的、在面试中不断被使用的技术问题，看看你有没有似曾相识的感觉。

下面这样的问题，可以在适合的场合下快速地使用，但是，通常来说却是不适合作为面试的“主要”问题的，换句话说，这些问题并不适合花大量的时间和候选人去讨论、分析和解决。

知识性的问题

有一些问题，是知识性的，这一类问题，考察的是知识，是记忆力，而不是任何一项工程师的核心能力。举例来说，有这样的问题：

> “对于 OA 系统，我们该怎样配置 Tomcat？”

为什么说这个问题不好呢？这是因为这里抠了一个 Tomcat 配置的细节：

> 首先，如果候选人不了解 Tomcat，回答不了这个问题呢？是不是就可以认为这是一个负面的考察数据？
>
> 其次，就算候选人了解 Tomcat，但是具体配置项记不起来了呢？这是否又可以成为一个负面的考察数据？

这一类问题，我们简单地称其为“知识性问题”，在技术面试中，我们通常就应该避免。**因为一方面它不具备普适性，另一方面它又具备太强的随机性。**

不具备普适性，指的是候选人必须要了解特定的框架或者库，考虑到优秀的候选人的知识背景有所区别，所以这一要求过于武断；而太强的随机性，指的是候选人是否知道这一个知识点，随机性太强，而并不能反映候选人的技能水准。

另外，**有一些问题并不像上面的例子那么直接，但是本质上有类似性。特定编程语言、框架和库限定下的问题就是其中之一**，比如：

> “请实现 C++ 的 atoi 算法。”

这个问题，如果团队就是要求候选人必须具备 C++ 技能，那没什么问题；但是对于大多数团队来说，情况并非如此，那么这个问题就对平时以 Java 为主要编程语言的候选人不太公平了。这也是为什么，很多大厂对于面试流程的培训中，有一条指导原则：不限制候选人使用的编程语言。

还有一些问题，可能有一些隐含的知识，或者说“背景知识”前提蕴藏在问题中，而这样的知识从面试角度来看，又不是我们关心的。

这一类问题需要注意，可以问，但这部分背景知识最好是众所周知的“常识”，倘若候选人不知道，那就应该最直接快速地告诉他，而不应该把时间浪费在它上面。比如说：

> “请设计一个算法，把十进制的数字，用罗马数字来表示。”

你看，这看似是一个挺好的问题，可假如说候选人不知道罗马数字的书写规则，是不是就卡壳了？而罗马数字的书写规则是我们对软件工程师候选人考察的内容吗？显然不是。

你可能会问，那么假如候选人知道这个规则，是不是就可以问了？

原则上没错，可前面已经提到了，我们需要的问题是具有“普适性”的，这样我们才能**把这个问题频繁地拿出来问，积累足够多的、可以用于比较的数据**。好问题千千万，何必死磕这一类呢？

你还可能会问，那我要是告知候选人这个规则，是不是这就变成一个好问题了呢？说得好，可答案依然是否定的，至于原因，我先卖个关子，你不妨先想一想。

过于常见的问题

这一类问题，我想应该很容易理解吧，但是我们却经常见到。举例来说，算法题有：

> “请对二叉树做一次先序遍历。”

系统设计问题有：

> “请设计一个 URL 短网址系统。”

我们考察候选人，是希望得到真实的数据，而被使用太多遍的问题，很有可能就是候选人准备好了的，这就让考察的真实性大打折扣了。诚实的候选人会告诉你，这题我做过了，但你不能奢望每个候选人都如是操作。

每一个面试官，随着经验与时间的积累，自己应该有那么两三个“杀手锏”问题。这样的问题经得起多次面试中询问的考验，并且积累了足够的数据，而更重要的是，这样的问题要具备一定的“独特性”，是候选人通常准备不到的。

另外，这里我必须要强调一下，我们这里是指问题本身不要过于常见，而不是说，“问题类型”和背后的“原理”不要过于常见。

相反，我们恰恰就是要通过“新”问题考察候选人的“旧”能力。这就是为什么我们说，问题是多变的，但套路是永恒的。

因为这些套路恰恰就是，软件工程师得以在问题的千变万化中，依然能够游刃有余地分析和解决它们的原因。比方说，URL 短网址系统的设计问题都被考烂了，但是背后的系统设计的考察要素，比如对于网络协议的理解、系统容量的估计、存储技术的选型等等，都是非常好的具体考察点。

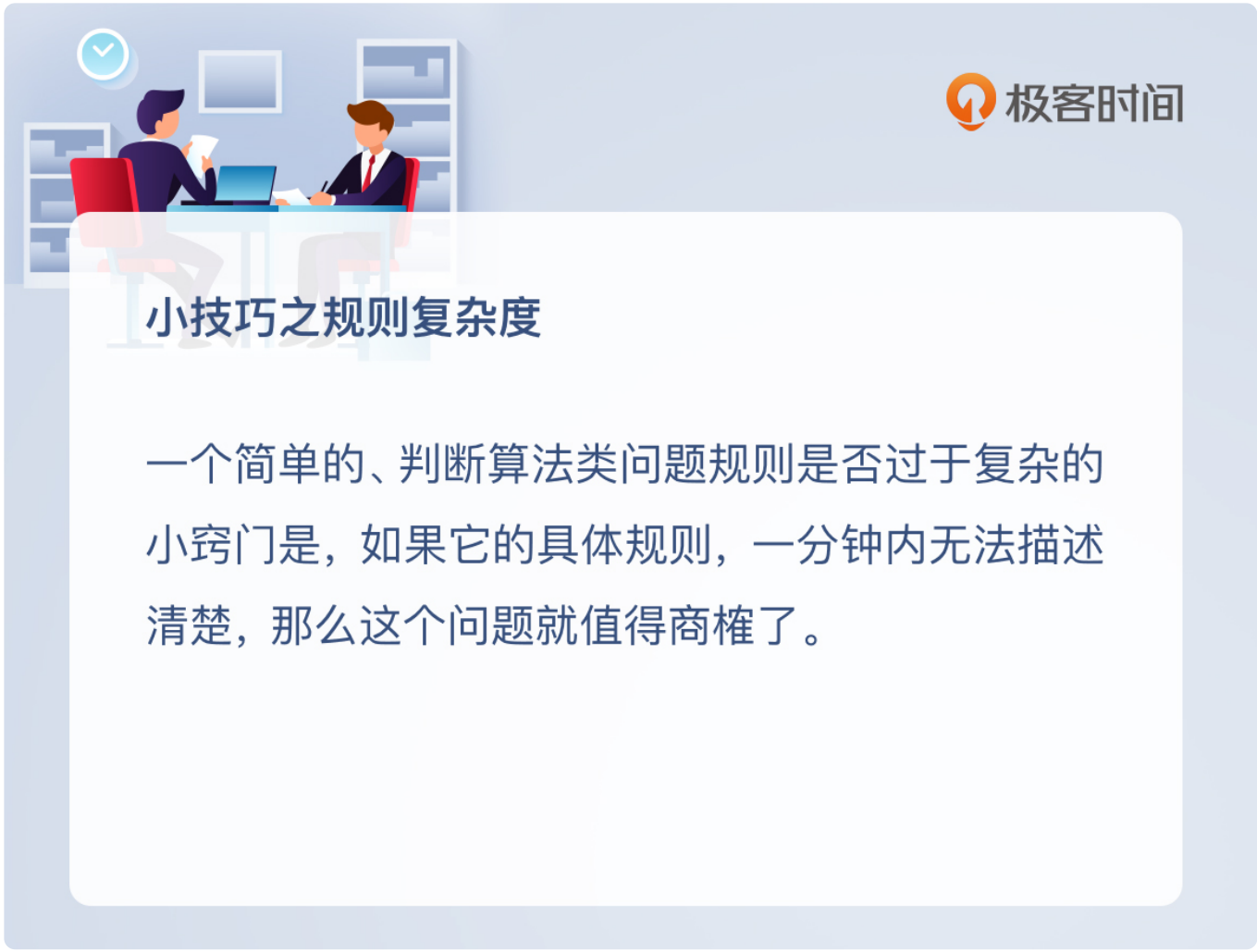
规则过于复杂的问题

再回到前面提到的罗马数字的问题：

> “请设计一个算法，把十进制的数字，用罗马数字来表示。”

为什么即便告知候选人规则，这依然不是一个好问题呢？因为这就可能是一个“规则过于复杂”的典型例子。因为如果候选人对罗马数字一无所知，那么把罗马数字的规则前前后后都讲清楚，怎么也得花费十多分钟的时间。

因此这样的问题，往往就不是一个好问题，毕竟，我们要把宝贵的时间尽量地分配到考察的重点内容上。



当然，这样的问题也不是不能“抢救”一下的。同样是考察代码层面的能力，如果我们给这个问题加一个小小的限制，比如，我们只考虑二十以内的数字，我们假设输入都是正确的等等，规则就会简单不少，那么，候选人就可以把精力集中在核心实现上。

需要说明的是，**复杂问题的简化途径，和考察项是密切相关的。**如果你就要考察候选人，是不是能够对各种场景考虑周全，那么就不要假设输入都是正确的，而让候选人自己去进行必要的输入判断和错误处理。

技术问题的设计原则

好，上面我们谈到了几个糟糕的技术问题，这是反面典型。接下来我们就从正面来谈谈，到底应该把握怎样的原则去设计和选择技术问题，以及到底怎样的问题才是好问题吧。

和考察角度、考察重点保持一致

不知你是否还记得，我们之前第 02 讲里，关于制定计划的内容，面试内容，就是要和考察角度、考察重点保持一致的。我们的目标，是和候选人一起来解决这个技术问题，在这

个过程中，我们可以围绕它来获得有效考察数据，从而便于我们在面试后做出决策。

举个例子，某面试官要对候选人进行面试考察，重点是“实际问题的解决以及对系统的理解”。于是，他决定使用一个“网约车系统的设计”问题来作为主要问题，在面试中和候选人一起讨论。那么，从这里我们大致可以看出，面试官需要把握这两个要点：

实际问题的解决：面试官打算给出实际问题，关注候选人是不是能够逐步通过工程师的技能，把这个“网约车”（比如滴滴打车，这就是一个实际的系统）实际问题中最核心的部分，简化和抽象为软件问题，并加以解决。

对系统的理解：面试官打算在和候选人讨论清楚需求后，要求候选人设计一个软件系统来实现它，这个过程中重点关注候选人是不是有系统设计的技能，以及是不是能够根据问题的特点，做出合适的技术选择。

对于怎样执行这个要点把握，还是不够明确对不对？别急，我们接着看，我会在接下来的技术问题设计原则中，完整地把它描述出来。

从模糊到清晰，从实际到抽象

你读到这里，可能会问，从模糊到清晰可以理解，可是什么是从实际到抽象呢？难道不应该是从抽象到实际吗？只有落到实际才能够实现啊！

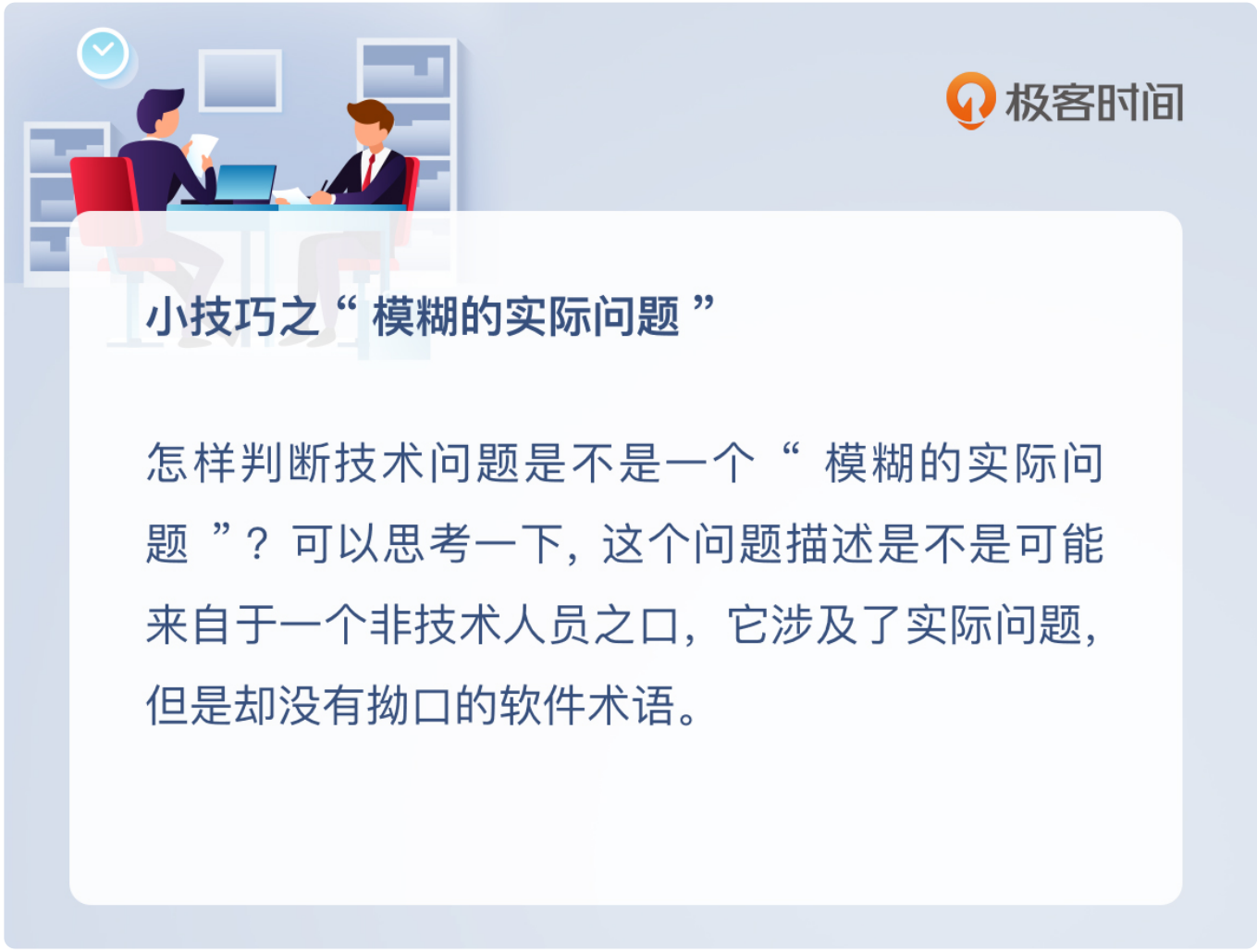
别急，我这就来做一个解释。从专栏一开始我就讲到，软件工程师，就是要做工程的，而做工程，从本质上说，就是要解决实际问题的。

这里说的从模糊到清晰，以及从实际到抽象，看似是两个过程，但其实它们恰恰是统一的：实际问题，往往就是模糊的，它可能来源于顾客的一句抱怨，用户的一个建议，或是一个不明不白的痛点；而软件途径可以解的问题，必须是清晰的、简单的，如果连软件的设计人员自己都说不清楚需求和逻辑，实现就无从谈起了，既然能够说清楚，这个描述肯定是经过了简化和抽象，去掉了细枝末节，只保留问题的核心。

因此，一个具体问题只会有某些核心的部分可以通过软件来解决，而余下的大部分只能是作为陪衬。**这一条原则，其实谈的是“深度”。**

这个对实际问题的简化和抽象的过程，正是软件工程师的重要能力，并且它还和实打实的经验密切相关。

我前面提到过，**纯算法题当然可以拿来问，但是我通常是不推荐的，因为考察面太窄，这样的问题本身就已经是一个经过抽象和简化了的数学问题了。**或者说，直接讨论这样的问题，就失去了将实际问题简化和抽象成软件工程可解的问题，这样一个非常有价值的过程。



我接着前面的例子来讲。面试官向候选人提出了这样的技术问题：

> “请你设计一个网约车系统。”

就这样简简单单一句话，有些候选人拿到问题以后，会不知所措，“天呐，我该从何下手？”没错，这个问题看起来就非常大，但这也是一个实际问题，工程师要去解决的，经常就是模糊的实际问题。

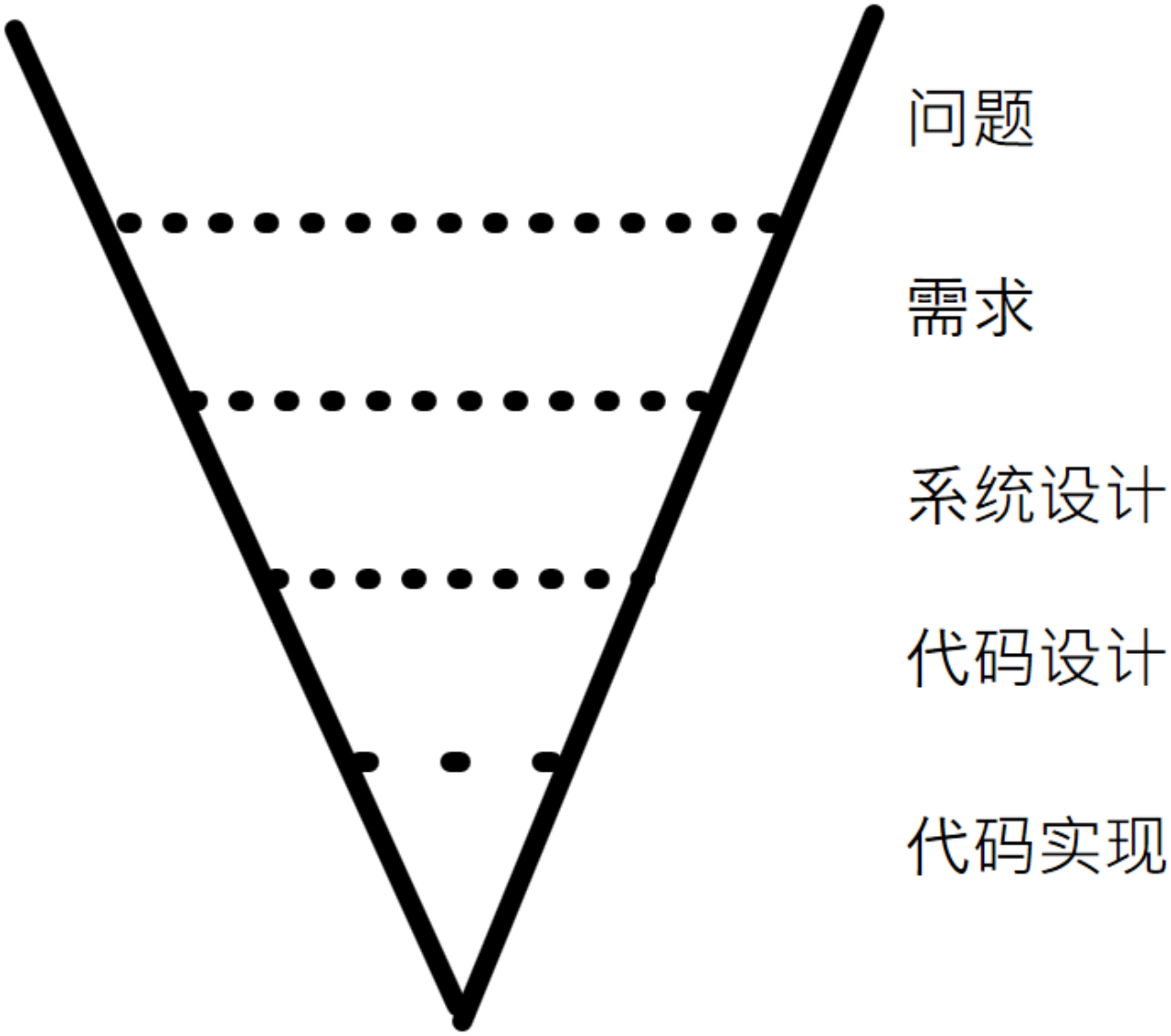
因此，我们希望和候选人一起，做出这样一个将问题从模糊到清晰，从实际到抽象的转化过程，以系统设计的考察路径为例：

1. 根据模糊的表述，明确我们要解决的核心问题是什么？网约车系统那么大，解决问题中，哪一类问题是我们最关注的。
2. 根据我们明确了的最关注的问题，列出相关的功能需求和非功能需求。需求可能包含很多，我们的目标是从中确定在面试时间内关心的部分。
3. 根据需求进行系统设计，包括从客户端、网络、service 到存储等等各层，有了大致的方案以后，我们的目标是从中筛选出我们最关心的一小部分展开讨论。
4. 如果还要考察代码层面，那么就从上述的设计中选取某一个组件，某一个机制，讨论代码层面的设计。
5. 根据代码层面的设计，择要实现，比如可以要求实现核心逻辑，核心数据结构等等。

从中你可以看出，整个思路是这样的：

> 问题 > 需求 > 系统设计 > 代码设计 > 代码实现

这不就是一个做迷你项目开发的过程吗？没错！虽然说，为了可操作性，每前进一步，都会缩小范围，纵深地考察候选人不同层次的技能。用一个图示来表示的话，正像一个漏斗：



需要说明的是，这就是一个最完整的典型系统设计考察路径，我们不一定要覆盖全部的步骤，但是我观察到许多面试官会覆盖前三步，或是将代码层面的考察，替代为深挖某一个组件或某一个机制的技术选型与 trade off。

不止一个考察角度，不止一个解

这里说的“不止一个考察角度”，指的是一开始同样模糊的描述，可以根据面试计划中的安排，逐步引导到某一个特定的考察角度上去。**这一条原则，其实谈的是“广度”。**

比方说，就上面提到的这个网约车系统的模糊问题，我们走的是一条系统设计考察的路径。

但是如果考察重点要求我们走一条数据结构和算法考察的路径，那也是可以的，比如从起点到终点的简单的寻路算法，落地到代码；我们还可以走面向对象设计考察的路径，设计

关键用例中涉及到的类，包括它们的结构和关联关系，等等。

跟随每一条路径，都一样可以逐步细化、抽象到一个具体的、软件可解的问题。

这里说的“不止一个解”，指的是在更低一级的层面，在细化到上述的具体的问题之后，解答不应该是唯一的，而应该存在多种不同的方法。比方说，寻路类的算法问题，我们有时可以用暴力回溯法来解，有时候可以用经过优化的动态规划方法来解，等等。

事实上，做面试官的一大乐趣正在此——可以和不同的候选人，一起讨论一些自己已经思考过、应用过的技术问题，看到不同人新颖的思路，有些很巧妙，有些很周全。

总结与思考

好，今天我们通过正反两个角度，先从反面介绍了几个糟糕的技术问题典型，再从正面结合例子讲解了我们该把握怎样的技术问题设计原则。

这些设计原则中，我想再谈一谈“从模糊到清晰，从实际到抽象”这一点。我们要看到的是，并不是所有的候选人都可以非常顺利地将一个模糊问题逐步细化清晰的，我们自然也不能复制同一套流程。

我的建议是，**越是软件经验丰富的候选人，我们就越可以给一个较为模糊的问题，尽量让他来主导这个分解和细化的过程；**而对于校招生，我们则可以给一个相对清晰具体的需求。当然，在这个过程中，如果发现难度过高，我们可以积极地给出提示，或者帮助他完成部分过程，这都是在合理范围内的。

好，今天的内容就是这些，如果你觉得在大方向上有数了，但是还不是特别有谱，那也别着急，因为在下一讲，我会继续讲解技术问题的设计技巧，以及实践中的注意点。在该过程中，我会将例子完全地展开。

最后，我想和你交流一下，你能否谈一谈，在你的面试经历中，是否遇到过糟糕的技术问题，而它们又为什么糟糕呢？

好，我是四火，我们下一讲见。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | 制定计划：好的计划是成功的一半

下一篇 04 | 问题设计（下）：五个技巧助攻技术问题设计

精选留言 (4)

写留言

楼下小黑哥
2021-04-20

之前有次面试，面试官提问，简历上某项目系统如何设计的。那我根据项目巴巴拉拉讲了一堆了，然后被面试官直接很强势提出你这么设计不行，你不觉得有XXX问题等等。
那听到面试官这么讲，我也不服，那我就按照我们系统一些现状，历史，然后描述系统为什么这么设计。然后针对他提出一些问题，我们是如何设计的。
但是这个过程感觉面试官不信服我说的设计，这就导致整个过程我们一直在对抗，面到...

展开

作者回复: 这个面试官也许就是这样的风格，制造冲突，然后看你的回应。但是我是不推荐这样的方式的，这就好像工作中一样，非得“不打不相识”吗？有些候选人能接受这一点，有些则不然。

当然，需要说明的是，是不是通过是一个综合考量，而不一定是单纯地根据这个产生冲突的事件而决定的。

1

璀璨星空
2021-04-19

面试实际相互考察的一个过程，优秀的工程师在面试过程中会考察面试官，能力"差"的面试者都是单向考虑问题，我前前后后面试过几十家公司，相对来说面试体验好的还是大厂，因为它有既定好的一整套面试流程，中小型公司基本都是今天提到的反面角度(占70%)吧，很容易错失优秀的工程师

展开

作者回复: 了解。这里所说的反面例子，都是我觉得比较常见的问题。

1

亚林
2021-04-18

这个行业，不管是项目经理，技术工程师，还是产品经理等的核心能力就将模糊的实际问题抽象成清晰的工程问题

展开

1

lihp

mip
2021-04-17

S 招聘人员主要是C/C++，笔试题中有一道题是关于C语言宏的用法，每次面试这道题的时候都会问：宏和函数有什么区别。
我觉得这是一个糟糕的问题，与特定的知识点相关，而且是书面上的常规介绍，只要是认真看过书或者看过C/C++面试指南的人都能说出基本概念，只能作为技术领域-编程语言-C的知识掌握情况，非C语言基本上无解，与能力无关，与思考方式无关。

展开

作者回复: 这个问题是一个基础知识问题，如果职位需求明确就是要C语言的功底，那么确实可以问，否则就不太适合。

1